Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Фізико-Технічний інститут

КРИПТОГРАФІЯ
КОМП’ЮТЕРНИЙ ПРАКТИКУМ №2

Виконала: Студентка 3-го курсу
Групи ФБ-93
Пономаренко Олександра Сергіївна

Київ 2021

Криптоаналіз шифру Віженера

Мета:

Засвоєння методів частотного криптоаналізу. Здобуття навичок роботи та аналізу потокових шифрів гамування адитивного типу на прикладі шифру Віженера.

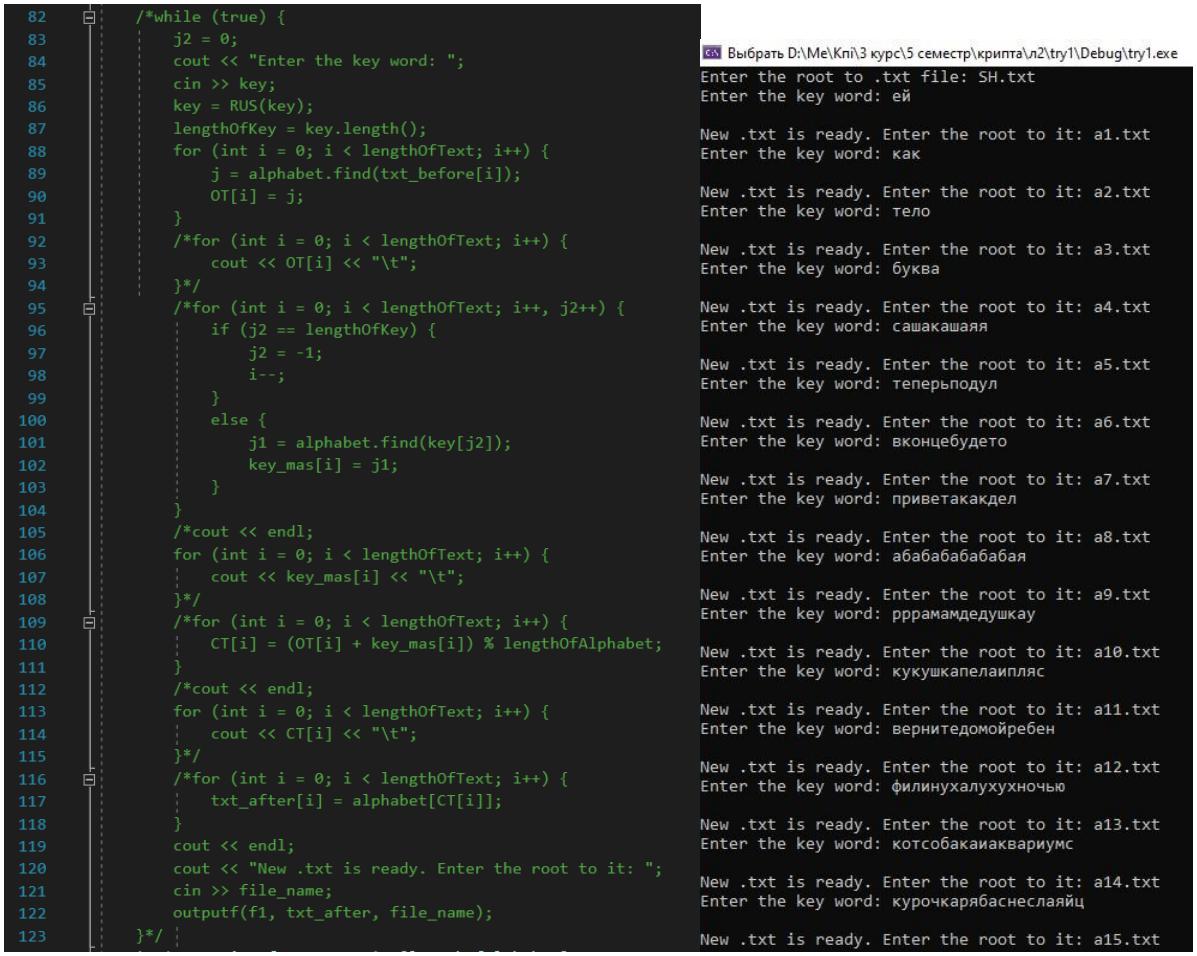
Завдання до виконання:

0. Уважно прочитати методичні вказівки до виконання комп'ютерного практикуму.

1. Самостійно підібрати текст для шифрування (2-3 кб) та ключі довжини r = 2, 3, 4, 5, а також довжини 10-20 знаків. Зашифрувати обраний відкритий текст шифром Віженера з цими ключами.
2. Підрахувати індекси відповідності для відкритого тексту та всіх одержаних шифртекстів і порівняти їх значення.
3. Використовуючи наведені теоретичні відомості, розшифрувати наданий шифртекст (згідно свого номеру варіанта).

Виконання роботи:

1)Для шифрування обрала текст, що знаходиться у SH.txt.



```
/*while (true) {
    j2 = 0;
    cout << "Enter the key word: ";
    cin >> key;
    key = RUS(key);
    lengthOfKey = key.length();
    for (int i = 0; i < lengthOfText; i++) {
        j = alphabet.find(txt_before[i]);
        OT[i] = j;
    }
    /*for (int i = 0; i < lengthOfText; i++) {
        cout << OT[i] << "\t";
    }*/
    /*for (int i = 0; i < lengthOfText; i++, j2++) {
        if (j2 == lengthOfKey) {
            j2 = -1;
            i--;
        }
        else {
            j1 = alphabet.find(key[j2]);
            key_mas[i] = j1;
        }
    }
    /*cout << endl;
    for (int i = 0; i < lengthOfText; i++) {
        cout << key_mas[i] << "\t";
    }*/
    /*for (int i = 0; i < lengthOfText; i++) {
        CT[i] = (OT[i] + key_mas[i]) % lengthOfAlphabet;
    }
    /*cout << endl;
    for (int i = 0; i < lengthOfText; i++) {
        cout << CT[i] << "\t";
    }*/
    /*for (int i = 0; i < lengthOfText; i++) {
        txt_after[i] = alphabet[CT[i]];
    }
    cout << endl;
    cout << "New .txt is ready. Enter the root to it: ";
    cin >> file_name;
    outputf(f1, txt_after, file_name);
}*/
```

```
Выбрать D:\Me\Kni\3 курс\5 семестр\крипта\л2\try1\Debug\try1.exe
Enter the root to .txt file: SH.txt
Enter the key word: ей

New .txt is ready. Enter the root to it: a1.txt
Enter the key word: как

New .txt is ready. Enter the root to it: a2.txt
Enter the key word: тело

New .txt is ready. Enter the root to it: a3.txt
Enter the key word: буква

New .txt is ready. Enter the root to it: a4.txt
Enter the key word: сашакашаяя

New .txt is ready. Enter the root to it: a5.txt
Enter the key word: теперьподул

New .txt is ready. Enter the root to it: a6.txt
Enter the key word: вконцебудето

New .txt is ready. Enter the root to it: a7.txt
Enter the key word: приветакакдел

New .txt is ready. Enter the root to it: a8.txt
Enter the key word: абабабабабабая

New .txt is ready. Enter the root to it: a9.txt
Enter the key word: пррамамдедушкау

New .txt is ready. Enter the root to it: a10.txt
Enter the key word: кукушкапелаипляс

New .txt is ready. Enter the root to it: a11.txt
Enter the key word: вернитедомойребен

New .txt is ready. Enter the root to it: a12.txt
Enter the key word: филинухалухухночью

New .txt is ready. Enter the root to it: a13.txt
Enter the key word: котсобакаиакваривумс

New .txt is ready. Enter the root to it: a14.txt
Enter the key word: курочкарябаснеслаяйц

New .txt is ready. Enter the root to it: a15.txt
```

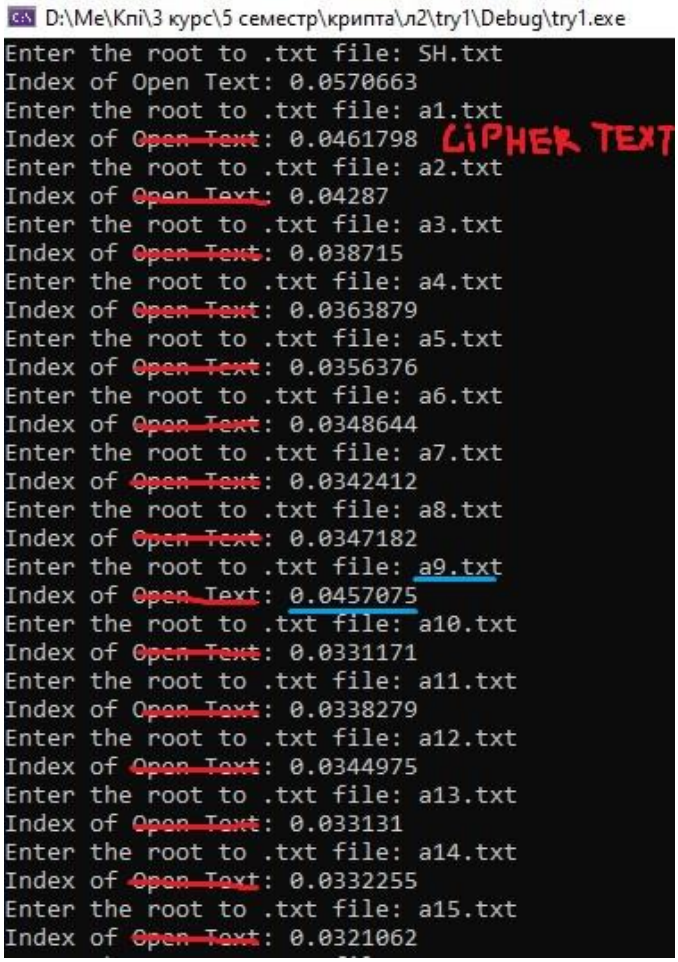
Далі в циклі зашифрувала ВТ різними ключами(відповідно з довжинами 2-5, 10-20). Записала усі ШТ в a1-15.txt(зі збільшенням індексу - збільшена довжина ключа).

2)У наступному циклі зчитувала усі отримані ШТ та обраховувала індекси відповідності для кожного з них.

```
while (true) {
    cout << "Enter the root to .txt file: ";
    cin >> file_name;
    txt_after = inputf(f, file_name);
    lengthOfText = txt_after.length();
    //cout << txt_after << endl;
    f_quantityOfSym(quantityOfSym, lengthOfAlphabet, alphabet, txt_after);
    index_OT = 0;
    index_OT = f_index_OT(lengthOfText, quantityOfSym, lengthOfAlphabet, index_OT);
    cout << "Index of ~~Open Text~~: " << index_OT << endl;
}
```

CIPHER TEXT

Отримала такі значення:



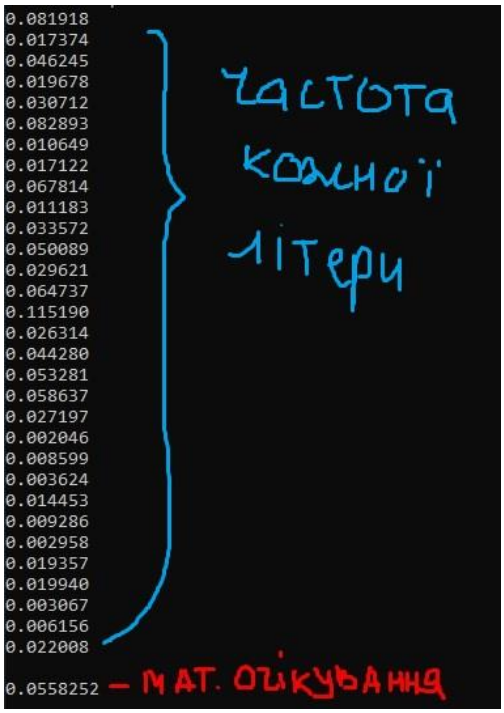
Дійсно, із збільшенням довжини ключа індекс зменшуються. Але в моєму випадку, у ШТ а9 стався стрибок. Я спочатку не зрозуміла, чому так, але потім подивившись на ключ для цього ШТ (а саме, абабабабабабая), зрозуміла, що це через нього. (А) - має індекс 0, тому зміни під час шифрування не відбуваються.

Код для першої частини знаходиться в try1.cpp

3) Варіант 9:

ШТ знаходиться у CT.txt

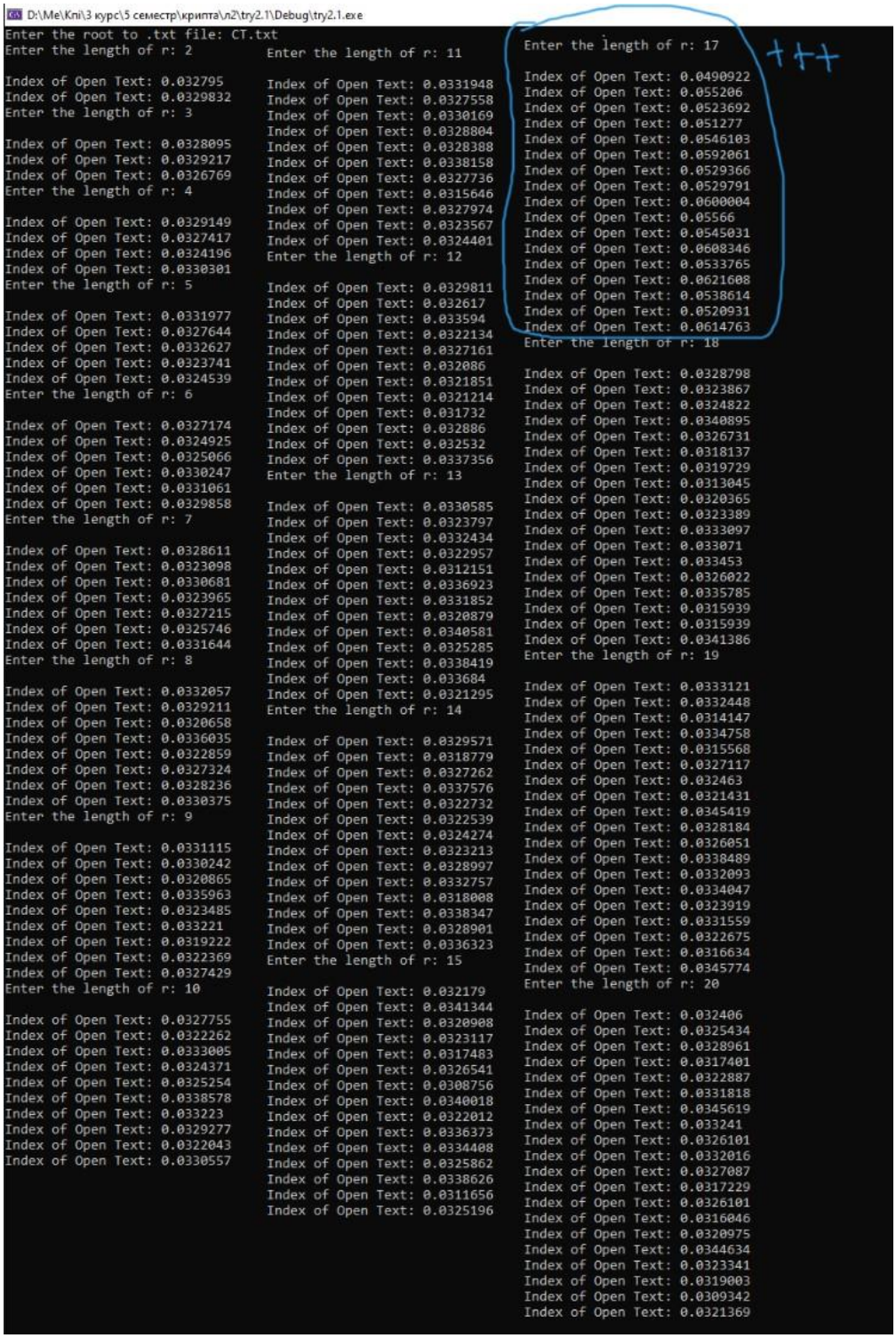
Для знаходження довжини ключа, нам потрібне теоретичне значення індексу відповідності (для знаходження використали отримані в першій лабораторній значення частот кожної літери):



Далі в циклі проводимо експеримент для різних довжин ключа, розбиваючи ШТ на блоки(довжина ключа=кі-сть блоків)

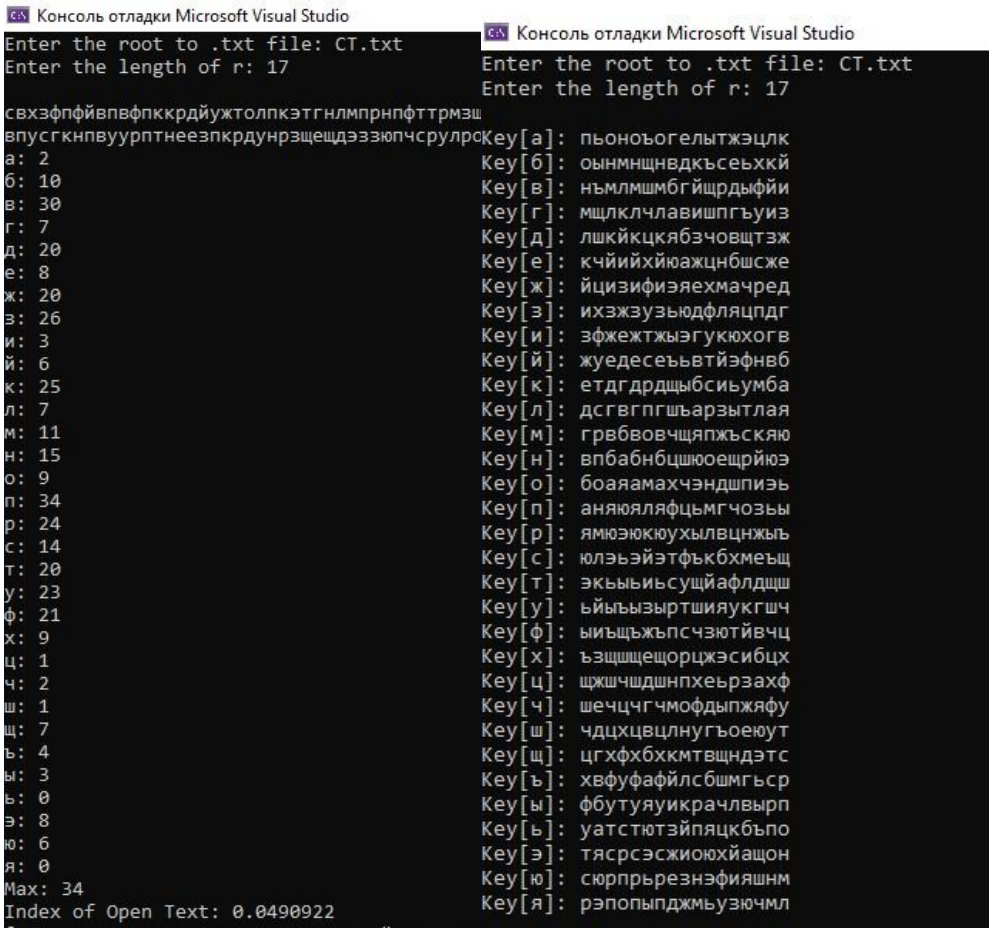
```
95      //знаходження довжини ключа
96      /*while (true) {
97          cout << "Enter the length of r: ";
98          cin >> lengthOfKey;
99          string* block = new string[lengthOfKey];
100         int extra;
101         //cout << txt_before << endl;
102         for (int rows = 0; rows < lengthOfKey; rows++) {
103             key = "";
104             extra = rows;
105             while (rows < lengthOfText) {
106                 key = key + txt_before[rows];
107                 rows = rows + lengthOfKey;
108             }
109             rows = extra;
110             block[rows] = key;
111         }
112         cout << endl;
113         int* quantityOfSym = new int[lengthOfAlphabet];
114         double index_OT = 0;
115         for (int i = 0; i < lengthOfKey; i++) {
116             //cout << block[i] << endl;
117             f_quantityOfSym(quantityOfSym, lengthOfAlphabet, alphabet, block[i]);
118             index_OT = f_index_OT(block[i].length(), quantityOfSym, lengthOfAlphabet, index_OT);
119             cout << "Index of Open Text: " << index_OT << endl;
120         }
121         delete[] block;
122     }*/
```

Продивляємося по всім значенням індексу, шукаючи найближче до математичного очікування. Саме при довжині 17, можемо бачити потрібні значення. Тому можна зробити висновок, що довжина нашого ключа дорівнює 17



```
D:\Me\Kni\3 курс\5 семестр\крипта\лл\try2.1\Debug\try2.1.exe

Enter the length of r: 21

Index of Open Text: 0.0353753
Index of Open Text: 0.0322437
Index of Open Text: 0.0332642
Index of Open Text: 0.0337853
Index of Open Text: 0.0316358
Index of Open Text: 0.0316575
Index of Open Text: 0.0307672
Index of Open Text: 0.0307889
Index of Open Text: 0.0324825
Index of Open Text: 0.0323306
Index of Open Text: 0.0308975
Index of Open Text: 0.0334162
Index of Open Text: 0.0343281
Index of Open Text: 0.0352835
Index of Open Text: 0.0325043
Index of Open Text: 0.0313752
Index of Open Text: 0.0348492
Index of Open Text: 0.0318312
Index of Open Text: 0.0326562
Index of Open Text: 0.0321786
Index of Open Text: 0.0354572
Enter the length of r: 22

Index of Open Text: 0.0320654
Index of Open Text: 0.0343172
Index of Open Text: 0.032445
Index of Open Text: 0.0303594
Index of Open Text: 0.0336773
Index of Open Text: 0.0341729
Index of Open Text: 0.033457
Index of Open Text: 0.031715
Index of Open Text: 0.0324786
Index of Open Text: 0.0306888
Index of Open Text: 0.0321922
Index of Open Text: 0.0350081
Index of Open Text: 0.0325502
Index of Open Text: 0.0341729
Index of Open Text: 0.0362491
Index of Open Text: 0.0326457
Index of Open Text: 0.0341252
Index of Open Text: 0.0309275
Index of Open Text: 0.0300445
Index of Open Text: 0.0337911
Index of Open Text: 0.0337434
Index of Open Text: 0.0323116
Enter the length of r: 23

Index of Open Text: 0.031504
Index of Open Text: 0.0325174
Index of Open Text: 0.0328031
Index of Open Text: 0.0309071
Index of Open Text: 0.0322317
Index of Open Text: 0.032881
Index of Open Text: 0.0343354
Index of Open Text: 0.0332446
Index of Open Text: 0.0316863
Index of Open Text: 0.0314785
Index of Open Text: 0.0358158
Index of Open Text: 0.0313746
Index of Open Text: 0.0343873
Index of Open Text: 0.0335563
Index of Open Text: 0.0334332
Index of Open Text: 0.0327007
Index of Open Text: 0.0334855
Index of Open Text: 0.0341918
Index of Open Text: 0.0322298
Index of Open Text: 0.0336686
Index of Open Text: 0.0319421
Index of Open Text: 0.0313403
Index of Open Text: 0.0323083

Enter the length of r: 24

Index of Open Text: 0.0314269
Index of Open Text: 0.0324022
Index of Open Text: 0.0327143
Index of Open Text: 0.0324306
Index of Open Text: 0.0313524
Index of Open Text: 0.0320901
Index of Open Text: 0.0342464
Index of Open Text: 0.0326009
Index of Open Text: 0.0313241
Index of Open Text: 0.033679
Index of Open Text: 0.0326292
Index of Open Text: 0.0344734
Index of Open Text: 0.0340479
Index of Open Text: 0.0321185
Index of Open Text: 0.0340195
Index of Open Text: 0.031239
Index of Open Text: 0.0338208
Index of Open Text: 0.0325441
Index of Open Text: 0.0312106
Index of Open Text: 0.0332818
Index of Open Text: 0.0310971
Index of Open Text: 0.0326576
Index of Open Text: 0.0321469
Index of Open Text: 0.0337641
Enter the length of r: 25

Index of Open Text: 0.0344363
Index of Open Text: 0.0311586
Index of Open Text: 0.0322615
Index of Open Text: 0.0313424
Index of Open Text: 0.0306684
Index of Open Text: 0.0343142
Index of Open Text: 0.032139
Index of Open Text: 0.0334564
Index of Open Text: 0.0310361
Index of Open Text: 0.0316181
Index of Open Text: 0.0316509
Index of Open Text: 0.0352328
Index of Open Text: 0.0326391
Index of Open Text: 0.0336889
Index of Open Text: 0.0316819
Index of Open Text: 0.0358813
Index of Open Text: 0.033689
Index of Open Text: 0.0325156
Index of Open Text: 0.0333493
Index of Open Text: 0.0329479
Index of Open Text: 0.0332258
Index of Open Text: 0.0313422
Index of Open Text: 0.0329787
Index of Open Text: 0.0314657
Index of Open Text: 0.0332257
Enter the length of r: 26

Index of Open Text: 0.0333831
Index of Open Text: 0.0326868
Index of Open Text: 0.0325541
Index of Open Text: 0.0335828
Index of Open Text: 0.0330518
Index of Open Text: 0.0332509
Index of Open Text: 0.0329523
Index of Open Text: 0.0313262
Index of Open Text: 0.0331513
Index of Open Text: 0.03272
Index of Open Text: 0.0336823
Index of Open Text: 0.031094
Index of Open Text: 0.0315253
Index of Open Text: 0.0327863
Index of Open Text: 0.0330518
Index of Open Text: 0.0359992
Index of Open Text: 0.0329882
Index of Open Text: 0.0306128
Index of Open Text: 0.0335234
Index of Open Text: 0.0331555
Index of Open Text: 0.0330551
Index of Open Text: 0.0345941
Index of Open Text: 0.0327206
Index of Open Text: 0.0325198
Index of Open Text: 0.0364676
Index of Open Text: 0.0323191

Enter the length of r: 27

Index of Open Text: 0.0346492
Index of Open Text: 0.0312529
Index of Open Text: 0.0308952
Index of Open Text: 0.0329692
Index of Open Text: 0.0306092
Index of Open Text: 0.033434
Index of Open Text: 0.034471
Index of Open Text: 0.0304662
Index of Open Text: 0.0324685
Index of Open Text: 0.0325759
Index of Open Text: 0.0334698
Index of Open Text: 0.0338631
Index of Open Text: 0.0329335
Index of Open Text: 0.0318073
Index of Open Text: 0.0326728
Index of Open Text: 0.0293551
Index of Open Text: 0.0312302
Index of Open Text: 0.0326006
Index of Open Text: 0.0330695
Index of Open Text: 0.0332858
Index of Open Text: 0.0329974
Index of Open Text: 0.0338628
Index of Open Text: 0.0342956
Index of Open Text: 0.0316991
Index of Open Text: 0.031086
Index of Open Text: 0.0323121
Index of Open Text: 0.0323843
Enter the length of r: 28

Index of Open Text: 0.031104
Index of Open Text: 0.0318423
Index of Open Text: 0.0309922
Index of Open Text: 0.034586
Index of Open Text: 0.0313787
Index of Open Text: 0.0347019
Index of Open Text: 0.0309922
Index of Open Text: 0.0325379
Index of Open Text: 0.0339677
Index of Open Text: 0.0309149
Index of Open Text: 0.0306444
Index of Open Text: 0.0365567
Index of Open Text: 0.0331563
Index of Open Text: 0.0330403
Index of Open Text: 0.0348951
Index of Open Text: 0.0306445
Index of Open Text: 0.0342768
Index of Open Text: 0.0322288
Index of Open Text: 0.0310695
Index of Open Text: 0.030799
Index of Open Text: 0.0339677
Index of Open Text: 0.0327698
Index of Open Text: 0.0308763
Index of Open Text: 0.0339677
Index of Open Text: 0.0328471
Index of Open Text: 0.0319969
Index of Open Text: 0.031881
Index of Open Text: 0.0331948

Enter the length of r: 29

Index of Open Text: 0.033731
Index of Open Text: 0.0327856
Index of Open Text: 0.0328678
Index of Open Text: 0.0336905
Index of Open Text: 0.02896
Index of Open Text: 0.0361152
Index of Open Text: 0.0336247
Index of Open Text: 0.0318396
Index of Open Text: 0.0348699
Index of Open Text: 0.0337492
Index of Open Text: 0.0327944
Index of Open Text: 0.0301792
Index of Open Text: 0.0303452
Index of Open Text: 0.0323792
Index of Open Text: 0.035119
Index of Open Text: 0.0327944
Index of Open Text: 0.0354096
Index of Open Text: 0.0339982
Index of Open Text: 0.0327529
Index of Open Text: 0.0327113
Index of Open Text: 0.0309264
Index of Open Text: 0.0337491
Index of Open Text: 0.033251
Index of Open Text: 0.0330019
Index of Open Text: 0.0305943
Index of Open Text: 0.0321717
Index of Open Text: 0.0323377
Index of Open Text: 0.035202
Index of Open Text: 0.0334586
Enter the length of r: 30

Index of Open Text: 0.0331296
Index of Open Text: 0.0319345
Index of Open Text: 0.0314915
Index of Open Text: 0.03096
Index of Open Text: 0.0308272
Index of Open Text: 0.033396
Index of Open Text: 0.0310044
Index of Open Text: 0.0332189
Index of Open Text: 0.0315359
Index of Open Text: 0.0353448
Index of Open Text: 0.0336176
Index of Open Text: 0.0330418
Index of Open Text: 0.0341933
Index of Open Text: 0.0306058
Index of Open Text: 0.0334403
Index of Open Text: 0.031713
Index of Open Text: 0.0364964
Index of Open Text: 0.0312702
Index of Open Text: 0.0328645
Index of Open Text: 0.0322002
Index of Open Text: 0.0308272
Index of Open Text: 0.0304285
Index of Open Text: 0.0337061
Index of Open Text: 0.0323774
Index of Open Text: 0.0332632
Index of Open Text: 0.0336679
Index of Open Text: 0.0318795
Index of Open Text: 0.0323713
Index of Open Text: 0.0319689
Index of Open Text: 0.0321477
```

Знаходимо у кожному блоці літеру, що найчастіше зустрічається:



І далі просто за схемою шифру Цезаря знаходимо можливі ключі. В даному варіанті просто обрати за найчастішу літеру (О) було недостатньо. Тому треба проводити частотний аналіз.

Дивлячись на ключ key[о], я помітила слово эндшпиэь. Було не складно догадатися, що насправді там эндшпиль. Але далі вже пішли складнощі. Розшифрувавши ШТ за допомогою key[о], отримала незв’язний текст, де на кожному 10-ому місці (ключ довжини 17) було зрозуміле слово, або навіть декілька, отже частина ключа эндшпиль була вгадана правильно. Далі дивлячись на неповноцінний ВТ намагалася закінчити або виправити слова, що вже були зрозумілими. Таким чином я і дійшла до значення ключа: войнамагаэндшпиль (key.txt).

Отриманий ВТ знаходиться в ОТ.txt. Код для другої частини лабораторної знаходиться в try2.1.cpp

Висновок:

За цю лабораторну роботу ми дізналися більше про поліалфавітні підстановки, а саме про шифр Віженера. Навчилися методу розшифрування для відповідних ШТ та практично продивилися залежність значення індексу відповідності від довжини ключа шифрування.